# اصول پردازش تصویر

نیم‌سال اول ۱۳۹۹-۱۴۰۰

مدرس: دکتر مصطفی کمالی تبریزی

دانشکده علوم ریاضی

## تمرین سری چهارم ـ سوال اول

نام و نام‌خانوادگی: سیدعلیرضا خادم

شماره دانشجویی: ۹۷۱۰۰۳۹۸

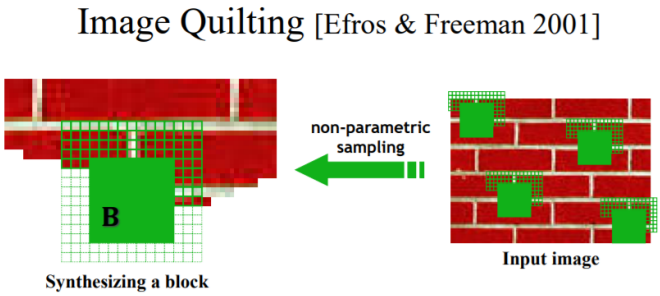
زمان حدودی اجرا: ۹۰ ثانیه

## موارد لازم.

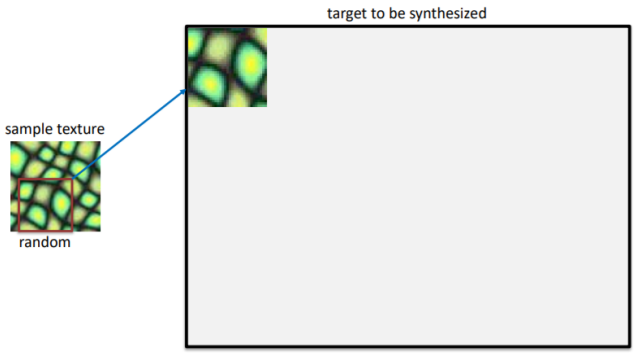
برای اجرا لازم است تا تصویر texture1.jpg در مسیر /EX4_Q1 قرار داشته باشد. همچنین در پیاده‌سازی این سوال از کتابخانه‌های cv2 و numpy استفاده شده است که قبل از اجرا بایستی این کتابخانه‌ها روی سیستم شما نصب باشد.

## روند کلی حل.

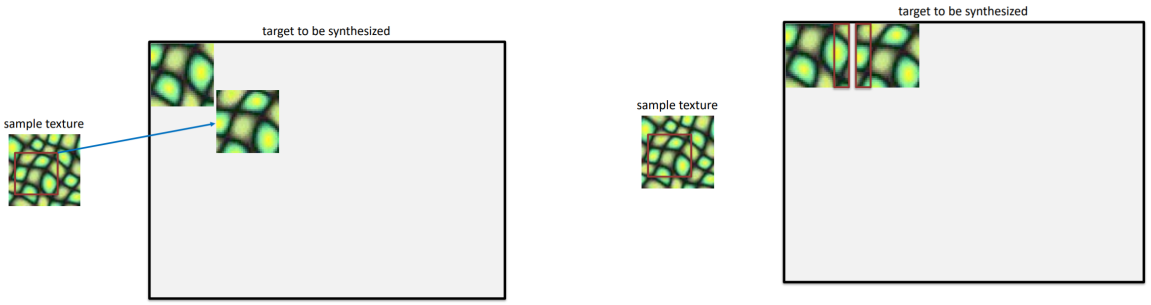
هدف سوال ساخت یک تصویر با اندازه ۲۵۰۰ پیکسل در ۲۵۰۰ پیکسل است که بافت آن مشابه بافتِ تصویر texture1.jpg در مسیر /EX4_Q1 باشد. برای انجام این کار ۲ ایده اولیه وجود داره : ۱. resize کردن سمپل داده شده برای تولید عکس با سایز بزرگتر ۲. build probability distribution. هر یک از این روش‌ها معایب خودشان رادارند و این معایب به اندازه‌ای در نتیجه نهایی تاثیرگذار است که باعث می‌شود برای حل این سوال سراغ این روش‌ها نرویم. ما در این جا از روش Image Quilting برای حل سوال استفاده می‌کنیم. ایده کلی این روش براساس این مشاهده است که "پیکسل‌های همسایه همبستگی زیادی با هم دارند". بر اساس این مشاهده، در روش Image Quilting تصویر نهایی را به‌جای این که پیکسل به پیکسل بسازیم، بلوک به بلوک می‌سازیم. کلیات این ایده را در تصویر زیر می‌توانید مشاهده کنید.



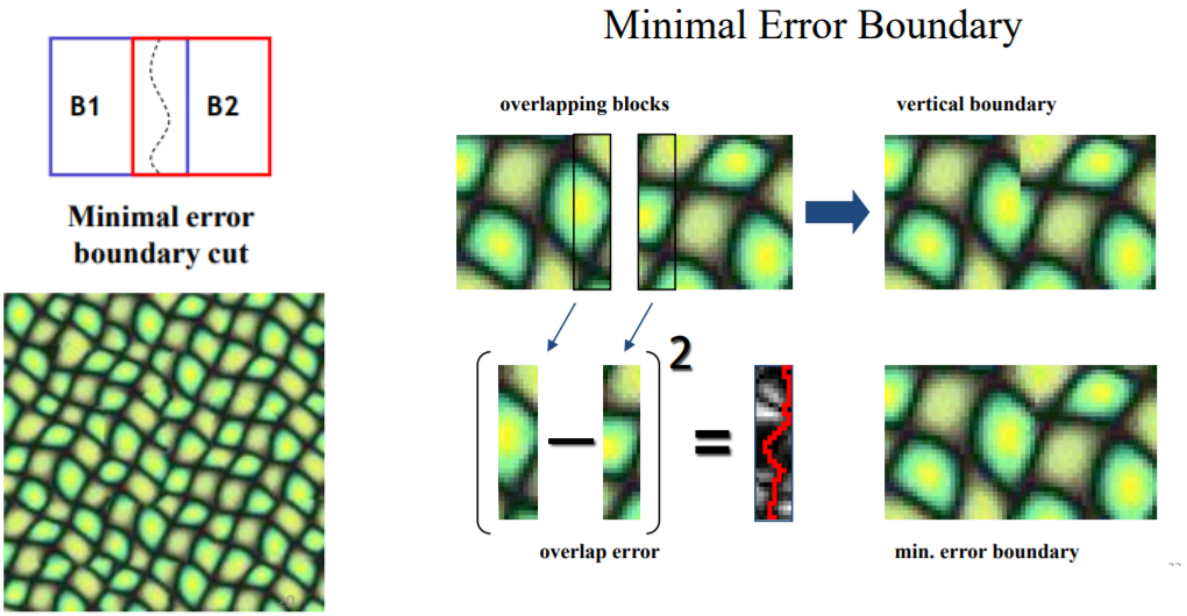
کاری که که انجام می‌دیم به این صورت است که در ابتدا یک block از تصویر sample را به صورت رندوم برمی‌داریم و در گوشه بالا سمت چپ تصویر target قرار می‌دهیم. (مشابه شکل زیر)



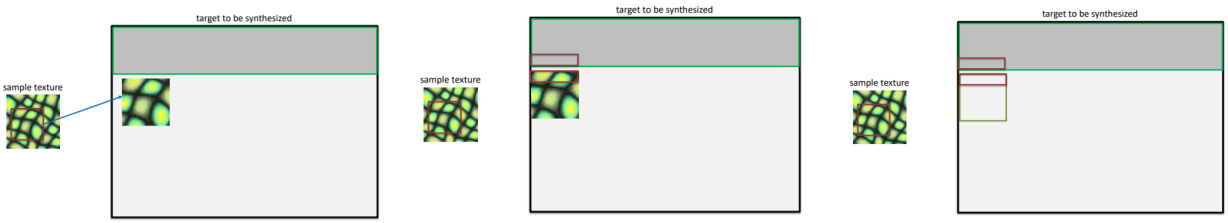

در ادامه یک patch جدید از تصویر sample انتخاب می‌کنیم، اما نه کاملا به صورت رندوم، بلکه به این صورت که می‌خواهیم این patch جدید به گونه‌ای انتخاب بشود که سمت چپ آن تا حد زیادی مشابه سمت راستِ patchای که در مرحله قبل در تصویر target قرار داده‌ایم، باشد؛ بنابراین یک نوار از سمت راست patchِ گام قبل در نظر می‌گیریم (مشابه عکس زیر) و همه patchهای به ابعاد patchِ گام قبل را که سمت چپشان شبیه به سمت راستِ patch گام قبل است را با یکی از روش‌های تمپلیت متچینگ (ما از روش cv.TM_CCORR_NORMED استفاده کردیم )پیدا می‌کنیم و از میان patch هایی که بیشترین شباهت را به patchِ گام قبل دارند، یکی را به صورت رندوم انتخاب می‌کنیم.




بعد از اینکه patch جدید را انتخاب کردیم، در قسمتی که patch جدید و patch قبلی overlap دارند با استفاده از dynamic programing ، Minimum Cut را طبق تصاویر زیر اعمال می‌کنیم. سمت راستِ کات را از patch جدید و سمتِ چپ کات را از تصویر target برمی‌داریم.




روندی که در بالا توضیح داده شد را ادامه می‌دهیم تا سطر اول از تصویر target ساخته شود. حال به سراغ سطر دوم می‌رویم. اولین patch از سطر دوم را که می‌خواهیم اضافه کنیم، باید به این صورت عمل کنیم که یک نوار از بالایِ این patch با یک نوار از پایینِ patchِ بالایی آن (مشابه عکس زیر) تا حدی خوبی مشابه باشند. و Min cut را توی این نوار اعمال می‌کنیم.

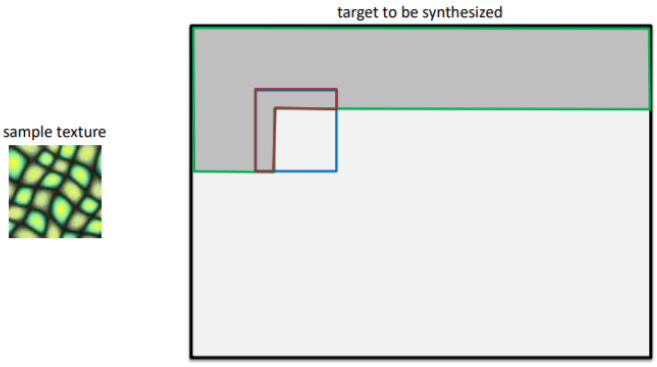



در ادامه‌ی ساختِ سطر دوم، overlapای که در نظر می‌گیریم تا بر اساس آن patch جدید را انتخاب کنیم به صورت L شکل در نظر می‌گیریم(مشابه تصویر زیر). و در هر یک از نوارهای بالا و سمت چپ Min cut را به دست می آوریم و جایی که این دو کات همگدیگر را قطع می‌کند را در نظر می‌گیریم و بالایِ کاتِ افقی و سمت جپ کاتِ عمودی را از تصویر target و پایینِ کاتِ افقی و سمتِ راستِ کات عمودی را از patch جدید می آوریم.

(ج) تقاطع دو کات

(ب) کات عمودی

(آ) کات افقی




روندی که در بالا برای سطر دوم توضیح دادیم را برای دیگر سطرها به صورت مشابه انجام می‌دهیم تا به طول کامل تصویر target ساخته شود.

## توضیح کد.

برنامه در مجموع حاوی ۲ فایل با فرمت py. می‌باشد که توضیحات هر فایل در پایین آمده است.

- **utilities.py**

**initialized_first_block(src_image, target_image, height_of_block, width_of_block)**

در این تابع ورودی‌های src_image ، target_image ، height_of_block و width_of_block را می‌گیرد و یک patch به ابعاد width_of_block × height_of_block به صورت تصادفی از تصویر src_image برمی‌دارد و در گوشه‌یِ بالا سمت چپ تصویر target قرار می‌دهد.

mask (آ)

**find_matching(src_image, patch, height_of_block, width_of_block)**

این تابع ورودی‌هایی که مشخص شده است را می‌گیرد و تصویر patch را داخل تصویر src_image با روش cv.TM_CCORR_NORMED سرچ می‌زند و نتیجه را در متغیر matching ذخیره می‌کند. در ادامه آن پیکسل‌هایی از matching که مقدار آن از matching.max() - 0001. بیشتر بود را به عنوان کاندیدِ آن قسمت‌هایی از src_image که بیشترین شباهت را به patch را دارد در متغیر locations ذخیره می‌کند. در نهایت هم یک قسمت از تصویر src_image را به عنوان یکی از قسمت که بیشترین شباهت را به patch دارد به صورت تصادفی از میان مختصات‌هایِ locations انتخاب کرده و به عنوان خروجی برمی‌گرداند.

**find_l_matching(src_image, patch, height_of_block, width_of_block, overlap)**

این تابع مشابه تابع find_matching است با این تفاوت که کاربرد آن در مواقعی است که اورلپ به صورت L شکل است. این تابع ورودی‌هایی که مشخص شده است را می‌گیرد و تصویر patch را داخل تصویر src_image با روش cv.TM_CCORR_NORMED و با در نظر گرفتنِ maskای که در ادامه تصویر آن آمده است سرچ می‌زند و نتیجه را در متغیر matching ذخیره می‌کند. در ادامه آن پیکسل‌هایی از matching که مقدار آن از matching.max() - 0001. بیشتر بود را به عنوان کاندیدِ آن قسمت‌هایی از src_image که بیشترین شباهت را به patch را دارد در متغیر locations ذخیره می‌کند. در نهایت هم یک قسمت از تصویر src_image را به عنوان یکی از قسمت که بیشترین شباهت را به patch دارد به صورت تصادفی از میان مختصات‌هایِ locations انتخاب کرده و به عنوان خروجی برمی‌گرداند.

**vertically_min_cut(first_patch, second_patch)**

این تابع برای محاسبه‌یِ کاتِ عمودی در ساختن سطر اول از target استفاده می‌شود؛ مواردی که اورلپ تنها شامل یک نوار عمودی می‌شود (این شرایط در سطر اول اتفاق می‌افتد).در این تابع first_patch و second_patch را به عنوان ورودی می‌گیرد و مین کات را محاسبه می‌کند به این صورت که مجذورِ تفاضل دو patch را در متغیر difference_of_squares ذخیره می‌کند و در ادامه با استفاده از dynamic programing یک مین کات عمودی را محاسبه می‌کند و در نهایت تصویری را به عنوان خروجی برمی‌گرداند که خود کات و سمت راست کات را از second_patch و سمت چپ کات را از first_patch قرار داده است.

**horizontally_min_cut(first_patch, second_patch)**

این تابع برای محاسبه‌یِ کاتِ افقی در ساختن ستون اول از target استفاده می‌شود؛ مواردی که اورلپ تنها شامل یک نوار افقی می‌شود (این شرایط در سطر اول اتفاق می‌افتد).در این تابع first_patch و second_patch را به عنوان ورودی می‌گیرد و مین کات را محاسبه می‌کند به این صورت که مجذورِ تفاضل دو patch را در متغیر difference_of_squares ذخیره می‌کند و در ادامه با استفاده از dynamic programing یک مین کات افقی را محاسبه می‌کند و در نهایت تصویری را به عنوان خروجی برمی‌گرداند که خود کات و بالای کات را از first_patch و پایین کات را از second_patch قرار

داده است.

**vertically_min_cut2(first_patch, second_patch)**

این تابع برای محاسبه‌یِ کاتِ عمودی در ساختن سطرهای دوم به بعد و ستون‌های دوم به بعد از target مورد استفاده قرار می‌گیرد. مواردی که اورلپ به صورت L شکل است.در این تابع first_patch و second_patch را به عنوان ورودی می‌گیرد و مین کات را محاسبه می‌کند به این صورت که مجذورِ تفاضل دو patch را در متغیر difference_of_squares ذخیره می‌کند و در ادامه با استفاده از dynamic programing یک مین کات عمودی را محاسبه می‌کند و در نهایت تصویری را به عنوان خروجی برمی‌گرداند که خود کات و سمت راست کات درآیه‌های صفر و سمت چپ کات را از first_patch قرار داده است. علاوه بر آن کات رو هم مشابه شکل زیر به عنوان خروجی برمی‌گرداند.

(آ) کات خروجی (ب) تصویر خروجی

**horizontally_min_cut2(first_patch, second_patch)**

این تابع برای محاسبه‌یِ کاتِ افقی در ساختن سطرهای دوم به بعد و ستون‌های دوم به بعد از target مورد استفاده قرار می‌گیرد. مواردی که اورلپ به صورت L شکل است.در این تابع first_patch و second_patch را به عنوان ورودی می‌گیرد و مین کات را محاسبه می‌کند به این صورت که مجذورِ تفاضل دو patch را در متغیر difference_of_squares ذخیره می‌کند و در ادامه با استفاده از dynamic programing یک مین کات افقی را محاسبه می‌کند و در نهایت تصویری را به عنوان خروجی برمی‌گرداند که خود کات و پایین آن درآیه‌های صفر و بالایِ کات را از first_patch قرار داده است. علاوه بر آن کات رو هم مشابه شکل زیر به عنوان خروجی برمی‌گرداند.

(آ) کات خروجی (ب) تصویر خروجی

**texture_synthesis(src_image, target_image, height_of_block, width_of_block, overlap)**

در این تابع با استفاده از توابعی که تا به اینجا توضیح داده شد و بر اساس روندی که در قسمت **روند کلی حل.** بررسی شد textureِ را تولید می‌کند.

- **q۱.py**

در این فایل ابتدا تصویر texture1.jpg را از مسیر /EX4_Q1/images لود می‌کنیم و در متغیر src_image قرار می‌دهیم و بعد آرایه ۳ بعدی $2700 \times 2700 \times 3$ target که همه درآیه‌های آن صفر است را تعریف می‌کنیم. در نهایت با فراخوانی تابع texture_synthesis روی ورودی‌هایی که مشاهده می‌کنید، target را با patchهای $150 \times 150$ که 55 پیکسل پهنایِ نوارهای overlap آنهاست، پرمی‌کنیم. در آخر هم نتیجه را با نام res01.jpg در مسیر /EX4_Q1/results ذخیره می‌کنیم.